FAT-Pointer based range addresses

AKILAN SELVACOUMAR, Heriot Watt University, UK

The increasing disparity between application workloads and the capacity of Translation Lookaside Buffers (TLB) has prompted researchers to explore innovative solutions to mitigate this gap. One such approach involves leveraging physically contiguous memory to optimize TLB utilization. Concurrently, advancements in hardware-level system security, exemplified by the Capability Hardware Enhanced RISC Instructions (CHERI) architecture, offer additional opportunities for improving memory management and security.

CHERI introduces capability-based addressing, a novel approach that enhances system security by associating capabilities with memory pointers. These capabilities restrict access to memory regions, thereby fortifying the system against various security threats. Importantly, the mechanisms implemented in CHERI for enforcing memory protection can also serve as accelerators for standard user-space memory allocators. By leveraging capability-based addressing, memory allocators can efficiently manage memory resources while ensuring robust security measures are in place.

ACM Reference Format:
Akilan Selvacoumar. 2018. FAT-Pointer based range addresses. *J. ACM* 37, 4, Article 111 (August 2018), 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

1.1 Background

2 FAT-POINTER BASED RANGE ADDRESSES

This experiment exploits the FAT pointer to enable faster

2.1 Overview

2.2 Range creation

2.3 Fragmentation

2.4 Allocation with huge pages

3 EVALUATION

4 LIMITATIONS

5 RELATED WORK

Efficient memory management, particularly in the context of Translation Lookaside Buffer (TLB) optimization [1], has been a focal point of research and development within computer architecture. Various techniques have been proposed to mitigate TLB-related bottlenecks and improve overall system performance.

Author's address: Akilan Selvacoumar, as251@hw.ac.uk, Heriot Watt University, UK.

0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

5.1 Huge Pages

This is used to map a very large region of memory to a single entry. This small/large region of memory is physically contiguous. Most implementations of huge pages are size aligned, For example for the x86 architecture the huge pages size are 4KB, 2MB and 1GB pages.

5.2 Segment

A segment can be viewed as mapping between contiguous virtual memory and contiguous physical memory. The property of a segment allows it to be larger than a page. Direct Segment <paper reference> allows the user to set a single segment for an application. Two registers are added to mark the start and end of the segment. Any virtual address within this region can be translated by adding the fixed offset between the virtual and physical address.

5.3 RMM

RMM introduces the concept of adding an additional range table. For large allocations RMM eagerly allocates contiguous physical pages. The following allocations creates large memory ranges that are both virtually and physically contiguous. RMM builds on the concept of Direct segment <paper reference> by adding offset to translate a virtual address to physical address. RMM compares address with range boundaries to decide which range it belongs to. RMM queries the range table ofter an L1 TLB miss.

5.4 FlexPointer

FlexPointer is based on the RMM<paper reference RMM> paper. FlexPointer does eagerly allocate pages which are physically contiguous and stores the ID to translate a virtual address to physical address on the remaining unused bits on the 64 bit virtual address. The paper contribution mentions shifting the TLB lookup to an earlier stage to improve latency of accessing the TLB entries. FlexPointer immediate queries the range TLB for translations rather than the RMM paper which waits for the L1 TLB miss.

6 SUMMARY

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

[1] Christopher Anderson and Sophia Drossopoulou. 2003. BabyJ: from Object Based to Class Based Programming via Types. *WOOD* 82, 7 (2003), 53–81.